Modelling Attack-Defense Trees using Timed Automata*

Olga Gadyatskaya[1], René Rydhof Hansen[2], Kim Guldstrand Larsen[2], Axel Legay[3], Mads Chr. Olesen[2], and Danny Bøgsted Poulsen[2],

[1] SnT, University of Luxembourg, Luxembourg
[2] Department of Computer Science, Aalborg University, Denmark
[3] INRIA Rennes – Bretagne Atlantique, France

Abstract. Performing a thorough security risk assessment of an organisation has always been challenging, but with the increased reliance on outsourced and off-site third-party services, i.e., "cloud services", combined with internal (legacy) IT-infrastructure and -services, it has become a very difficult and time-consuming task. One of the traditional tools available to ease the burden of performing a security risk assessment and structure security analyses in general is *attack trees* [19, 23, 24], a tree-based formalism inspired by *fault trees*, a well-known formalism used in safety engineering.
In this paper we study an extension of traditional attack trees, called *attack-defense trees*, in which not only the attacker's actions are modelled, but also the defensive actions taken by the attacked party [15]. In this work we use the attack-defense tree as a goal an attacker wants to achieve, and separate the behaviour of the attacker and defender from the attack-defense-tree. We give a fully stochastic timed semantics for the behaviour of the attacker by introducing *attacker profiles* that choose actions probabilistically and execute these according to a probability density. Lastly, the stochastic semantics provides success probabilitites for individual actions. Furthermore, we show how to introduce costs of attacker actions. Finally, we show how to automatically encode it all with a *network of timed automata*, an encoding that enables us to apply state-of-the-art model checking tools and techniques to perform fully automated quantitative and qualitative analyses of the modelled system.

1 Introduction

In the past few years, we have witnessed a rapid increase in the number and severity of security breaches, ranging from theft of personal information about millions of US government employees[1] to sophisticated targeted malware attacks on security vendors[2]. This problem is exacerbated by the fact that it has become

* Research leading to these results was partially supported by the European Union Seventh Framework Programme under grant agreement no. 318003 (TREsPASS).

[1] `https://www.opm.gov/cybersecurity/cybersecurity-incidents/`

[2] `http://usa.kaspersky.com/about-us/press-center/press-releases/duqu-back-kaspersky-lab-reveals-cyberattack-its-corporate-netwo`

difficult to perform an adequate *risk assessment* of an organisation's security stance, with many organisations relying on a complex mix of off-site third party IT-services, e.g., "cloud services" and internally supported IT services. One of the tools available to help structure risk assessments and security analyses is *attack trees*, recommended, e.g., by NATO Research and Technology Organisation (RTO) [20] and OWASP [22]. Attack trees [19, 23, 24] is a tree based formalism inspired by *fault trees*, a well-known formalism used in safety engineering. The formalism was initially introduced by Schneier [24] and given a formal definition by Mauw and Oostdijk [19]. Kordy et al. [16] provide a survey on attack trees and related formalisms. While basic quantitative analysis, i.e., a bottom-up computation for a single parameter (e.g., cost, probability or time of an attack), can be performed directly on attack trees [4], several proposals exist to extend the basic attack tree formalism in order to support better analysis. For example, Buldas et al. [6], Jürgenson and Willemson [14] introduced multi-parameter attack trees with interdependent variables; Dalton et al. [7] have proposed analysing attack trees as Generalized Stochastic Petri Nets; Arnold et al. [2] applied interactive Input/Output Markov Chains to enhance temporal and stochastic dependencies analysis in attack trees. Kumar et al. [17] have considered priced timed automata for analysis of attack trees. This work defines a translation for each leaf node and each gate in an attack tree into a priced timed automaton. The approach allows to translate the full attack tree into an automaton that can be analysed using the UPPAAL CORA model checker. The research community interest in attack trees has been recently reinvigorated by new techniques to automatically generate attack trees and attack-defense trees from socio-technical organizational models [11, 13], paving the way towards automating risk assessment.

Attack-defense trees are a notable extension of attack trees that include, besides attacker's actions, also defender's actions and model their interplay [3, 15]. This extended formalism allows capturing more detailed scenarios, and incorporating the defender's perspective into an analysis. For example, burglar-resistance classes for physical security mechanisms, such as doors and windows, define how much time an attacker equipped with certain tools needs to spend on the intrusion [25]. Explicit consideration of defenses in the analysis allows the domain experts to get a better picture of the scenario [4, 15]. Recently, Hermanns et al. [12] have created the attack-defense-diagrams formalism extending attacke-defense trees with trigger and reset gates, which allow expressing temporal behaviours. The work [21] likewise introduces a sequential gate to attack-defense trees and considers a two-player stochastic game interpretation of this.

Our paper introduces a framework for analysing complex temporal scenarios of interactions of attackers and defenders, beyond the expressiveness of classic attack-defense trees. For doing this we develop a modelling framework for expressing the temporal behaviour of the attacker with the formalism *networks of timed automata*. Unlike the work of [17] the attack-defense-tree is not encoded as a timed automata - instead it is encoded as a boolean formula, which the attacker wishes to become true. This encoding allows us to apply state-of-the-art model checking tools and techniques to perform fully automated analyses of the modelled system,

both qualitative (boolean) analysis and quantitative (probabilistic) analysis. The modelling framework is accompanied by an automatic translation script. The script reads an attack-defense-tree and outputs a UPPAAL [18] timed automata model which can subsequently be queried several questions: among these questions are "what is the probability that an attack succeeds within τ" and "what is the expected cost of the attacker within τ time units" for a specific behaviour of the attacker. Using UPPAAL-STRATEGO [10], a recent extension of UPPAAL, we are furthermore capable of finding an attacker that minimises the expected cost of an attack.

2 Attack Defense Trees

We will now define an attack-defense tree (Definition 1), along with the standard boolean semantics for such a tree. Thereafter a temporal semantics with time, cost and stochasticity is introduced. This temporal semantics is the first contribution of this paper.

Definition 1 (AD-tree). *An AD-tree over the attacker actions* $\mathtt{A}_a$ *and defender actions* $\mathtt{A}_d$ *is generated by the syntax*

$$t ::= p \mid t \wedge t \mid t \vee t \mid \sim t$$

where $p \in \mathtt{A}_a \cup \mathtt{A}_d$*. We denote by* $\mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$ *all AD-trees over* $\mathtt{A}_a$ *and* $\mathtt{A}_d$*.*

Let $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$, let $A \subseteq \mathtt{A}_a$ be the set of selected attacker actins and let $D \subseteq \mathtt{A}_d$ be the set of selected defender actions; then we inductively define $[\![t]\!]A, D$ as

- $[\![p]\!]D, A = \mathtt{tt}$ if $p \in A \cup D$, $\mathtt{ff}$ otherwise
- $[\![t_1 \wedge t_2]\!]D, A = ([\![t_1]\!]D, A) \wedge ([\![t_2]\!]D, A)$
- $[\![t_1 \vee t_2]\!]D, A = ([\![t_1]\!]D, A) \vee ([\![t_2]\!]D, A)$
- $[\![\sim t]\!]D, A = \neg([\![t]\!]D, A)$

As an example of an attack-defense-tree consider Fig. 1. This tree explains how an attacker may succefully remove an RFID-tag from a warehouse. Among the possible ways is infiltrating management and order a replacement tag. The example is lifted from [3].

To make attack-defense-trees well-formed, we follow Aslanyan and Nielson [3] and impose a type system on top of the abstract syntax of Definition 1 – in this system there are two types d and a corresponding to defender and attacker. The type system is captured in Fig. 2. The negation operator $\sim$ acts like the switch operator of Aslanyan and Nielson [3] and changes the type of the subtree. Unlike Aslanyan and Nielson, we do not have a normal negation operator: the reason is we only want an attacker (or defender for that matter) to do positive things i.e. the attacker should only do something beneficial for him. In the remainder we only consider well-formed trees according to this type-system and we restrict our attention to trees t where $t \vdash a$. The major interest of attack-defense trees is whether there exists a set of defense measures such that an attack can never occur.

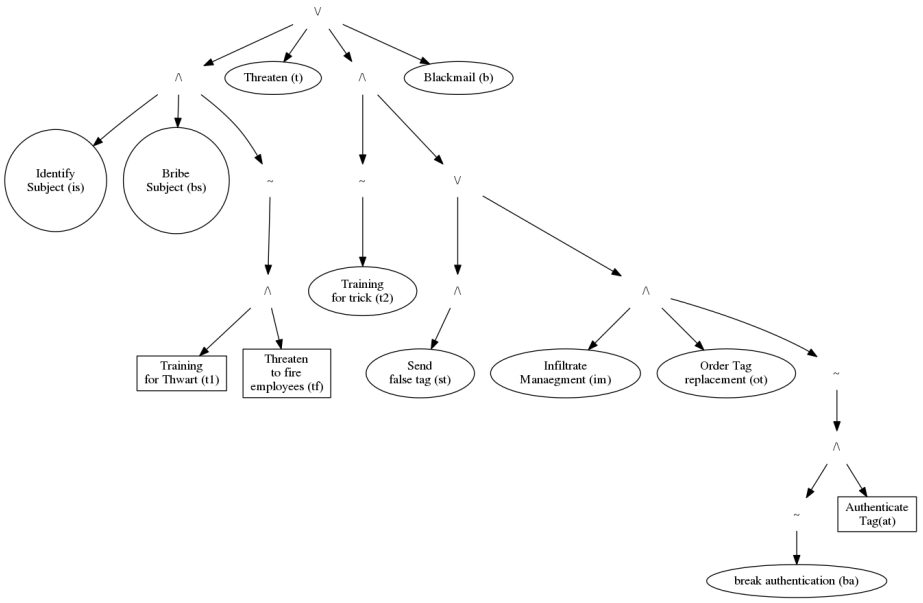


Fig. 1: An example of an attack-defense-tree. Square items correspond to defender's actions and circles to the attacker.

Question 1. For an attack-defense tree $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$, does there exist $D \subseteq \mathtt{A}_d$, such that for all $A \subseteq \mathtt{A}_a$, $[\![t]\!]D, A = \mathtt{ff}$?

This encapsulates our view that defense measures are selected ahead of time and fixed, while the attacker selects a set of attack measures. Our view is in accordance with the classical definition of attack-defense trees by [15]. Let λ be a symbol not in $\mathtt{A}_a$, which indicates that an attacker chooses to do no actions. We denote by $\mathtt{A}_a^\lambda$ the set $\mathtt{A}_a \cup \{\lambda\}$.

Definition 2. *Let $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$ be an AD-tree. The Attack-Defense-Graph over t is the tuple $\mathcal{G}^t = (\mathcal{V}, v^0, \rightarrow, \rightarrow_\neg, \dashrightarrow, F)$ where*

$$\frac{}{\mathtt{A}_d, \mathtt{A}_a \vdash p : a}, p \in \mathtt{A}_a \qquad \frac{}{\mathtt{A}_d, \mathtt{A}_a \vdash p : d}, p \in \mathtt{A}_d \qquad \frac{\mathtt{A}_d, \mathtt{A}_a \vdash t_1 : r \qquad \mathtt{A}_d, \mathtt{A}_a \vdash t_2 : r}{\mathtt{A}_d, \mathtt{A}_a \vdash t_1 \wedge t_2 : r}$$

$$\frac{\mathtt{A}_d, \mathtt{A}_a \vdash t_1 : r \qquad \mathtt{A}_d, \mathtt{A}_a \vdash t_2 : r}{\mathtt{A}_d, \mathtt{A}_a \vdash t_1 \vee t_2 : r} \qquad \frac{\mathtt{A}_d, \mathtt{A}_a \vdash t_1 : r}{\mathtt{A}_d, \mathtt{A}_a \vdash \sim t_1 : r^{-1}}, r^{-1} = \begin{cases} a & \text{if } r = d \\ d & \text{if } r = a \end{cases}$$

Fig. 2: Type system to make attack-defense trees well-formed

- $\mathcal{V} = 2^{\mathtt{A}_d} \times 2^{\mathtt{A}_a}$ *is a set of vertices containing currently true attacker and defender actions,*
- $v^0 = (\emptyset, \emptyset)$ *is the initial vertex,*
- $\rightarrow \subseteq (\mathcal{V} \times \mathtt{A}_a^{\lambda} \times \mathcal{V})$ *is a set of edges where* $((D, A), a, (D', A')) \in \rightarrow$ *if and only if* $D = D'$, $A' = A \cup (\{a\} \cap \mathtt{A}_a)$ *and* $a \notin A$,
- $\rightarrow_{\neg} \subseteq (\mathcal{V} \times \mathtt{A}_a \times \mathcal{V})$ *is a set of edges where* $((D, A), a, (D, A)) \in \rightarrow_{\neg}$ *if and only if* $a \notin A$
- $\dashrightarrow = \{(v^0, D, S) \mid D \in 2^{\mathtt{A}_d} \wedge S = (D, \emptyset)\}$ *is the "select defense" edges and*
- $F = \{(D, A) \in \mathcal{V} \mid [\![t]\!]D, A = \mathtt{tt}\}$ *is a set of final vertices.*

An attack-defense graph is essentially laying out all the possible steps an attacker may take to achieve a successful attack. Notice the edges in $\rightarrow_{\neg}$ correspond to trying to execute an atomic attack and failing. We allow this loop back as in this way we are able to model an attacker who selects what action to perform and an environment deciding whether that action succeeds.

For an attack-defense graph (ADG) $\mathcal{G}^t = (\mathcal{V}, v^0, \rightarrow, \rightarrow_{\neg}, \dashrightarrow, F)$ we write $v \overset{D}{\dashrightarrow} v'$ whenever $(v, D, v') \in \dashrightarrow$ and similarly we write $v \xrightarrow{a} v'$ ($v \xrightarrow{\neg a} v'$) if $(v, a, v') \in \rightarrow$ ($(v, a, v') \in \rightarrow_{\neg}$). An *attack-defense scenario* (ADS) for $\mathcal{G}^t$ is a sequence $\omega = v_0 D v_1 \alpha_1 v_2 \alpha_2 \ldots \alpha_{n_1} s_n \ldots$, where $v_0 = v^0$, for all i, $\alpha_i \in \{a, \neg a \mid a \in \mathtt{A}_a\} \cup \{\lambda\}$, $v_0 \overset{D}{\dashrightarrow} v_1$ and for all $j > 0$, $v_j \xrightarrow{\alpha_j} v_{j+1}$. We call ω a *successful ADS* if there exists j such that $v_j \in F$, denoted $\omega \vDash t$, and we call it a *failed ADS* if for all j, $v_j \notin F$, denoted $\omega \nvDash t$. We denote by $\Omega(t)$ all ADSs over t and furthermore let $\Omega^D(t) = \{\pi = v^0 \overset{D}{\dashrightarrow} v_0 \xrightarrow{a_1} \cdots \mid \pi \in \Omega(t)\}$ be all the ADSs initiated by the defender selecting defense measure D.

Lemma 1. *Let* $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$ *be an attack-defense-tree and let* $D \subseteq \mathtt{A}_d$*. If for all* $\omega \in \Omega^D(t)$, $\omega \nvDash t$ *then for all* $A \subseteq \mathtt{A}_a$ $[\![t]\!]D, A = \mathtt{ff}$.

Lemma 2. *Let* $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$ *be an attack-defense-tree and let* $D \subseteq \mathtt{A}_d$*. If there exists* $\omega \in \Omega^D(t)$, $\omega \vDash t$ *then there exists* $A \subseteq \mathtt{A}_a$ *such that* $[\![t]\!]D, A = \mathtt{tt}$.

In reality we wish to analyse the possible attacks after the defender has selected some defense measures. For this we remove the choice of defense measures from the ADG to get an attack graph (AG). Let $\mathcal{G}^t = (\mathcal{V}, v^0, \rightarrow, \rightarrow_{\neg}, \dashrightarrow, F)$ be the ADG for $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$; then the AG responding to $D \subseteq \mathtt{A}_d$ is the graph $(\mathcal{V}, v^{\mathcal{A}}, \rightarrow, \rightarrow_{\neg}, F)$ where $v^0 \overset{D}{\dashrightarrow} v^{\mathcal{A}}$. We denote this AG by $\mathcal{G}^t_D$. Due to Lemma 1 and Lemma 2 then question 1 is answerable by a pure reachability check on $\mathcal{G}^t_D$ for all $D \subseteq 2^{\mathtt{A}_d}$.

2.1 Adding Timed Behaviour

Intuitively speaking, an attacker observes the state of an ADG and choose an action. The attacker is memoryless and does, for instance, not remember how many times a specific attack has been attempted. The execution time of an action p_a is given by interval $[L_{p_a}, U_{p_a}]$, and thus an abstract timed attacker (Definition 3) is essentially a timed transition system.

Definition 3 (Abstract Timed Attacker). *Let $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$. An abstract timed attacker over the ADG $\mathcal{G}^t = (\mathcal{V}, v^0, \rightarrow, \rightarrow_\neg, \dashrightarrow, F)$ is a tuple (S, M, Ac) where*

- S *is a set of states,*
- $M : \mathcal{V} \to \mathsf{S}$ *maps vertices to attacker states, and*
- $Ac : \mathsf{S} \to 2^{\mathtt{A}_a^\lambda \times \mathbb{R}_{\geq 0}}$ *gives the possible actions and delays for an attacker, with the requirements that*
  - *if* $\mathsf{s} = M(v)$ *and* $(p_a, r) \in Ac(\mathsf{s})$ *then* $v \xrightarrow{p_a} v'$ *for some* v',
  - *if* $(p_a, t) \in Ac(\mathsf{s})$ *then* $L_{p_a} \leq t \leq U_{p_a}$ *and* $\{(p_a, t') | L_{p_a} \leq t' \leq U_{p_a}\} \subseteq Ac(\mathsf{s})$,
  - *if* $(\lambda, t) \in Ac(\mathsf{s})$ *then* $Ac(\mathsf{s}) = \{(\lambda, t') \mid t' \in \mathbb{R}_{\geq 0}\}$,
  - *if* $\mathsf{s} = M(v)$, $v = D, A$, *then* $(\lambda, 0) \in Ac(\mathsf{s})$ *if and only if* $v \in F$ *or* $A = \mathtt{A}_a$ *and*
  - *for all* $\mathsf{s} \in \mathsf{s}$, $Ac(\mathsf{s}) \neq \emptyset$

Let $\mathcal{G}^t = (\mathcal{V}, v^0, \rightarrow_a, \rightarrow_\neg, \dashrightarrow, F)$ be an ADG and let $\mathcal{A} = (\mathsf{S}, M, Ac)$ be an abstract timed attacker for $\mathcal{G}^t$. For $D \subseteq \mathtt{A}_d$, we denote by $\mathcal{G}^t_D|\mathcal{A}$ the transition system with state space $\mathcal{V} \times \mathsf{S}$, initial state $(v^{\mathcal{A}}, M(v^{\mathcal{A}}))$ and transition relation defined by the rules

- $(v, \mathsf{s}) \xrightarrow{p_a, t} (v', M(v'))$ if $(p_a, t) \in Ac(\mathsf{s})$ and $v \xrightarrow{p_a} v'$
- $(v, \mathsf{s}) \xrightarrow{\neg p_a, t} (v', M(v'))$ if $(p_a, t) \in Ac(\mathsf{s})$ and $v \xrightarrow{\neg p_a} v'$
- $(v, \mathsf{s}) \xrightarrow{\lambda, t} (v', M(v'))$ if $(\lambda, t) \in Ac(\mathsf{s})$ and $v \xrightarrow{\lambda} v'$.

A timed attack over $\mathcal{G}^t_D|\mathcal{A}$, $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$ is a sequence $v_0 d_0 \alpha_0, v_1 d_1 \alpha_1 \ldots$, where $v_0 = v^{\mathcal{A}}$, for all i, $d_i \in \mathbb{R}_{\geq 0}$, $\alpha_i \in \{p_a, \neg p_a \mid p_a \in \mathtt{A}_a\} \cup \{\lambda\}$ and there exists a sequence of states and transitions $(v_0, M(v_0)) \xrightarrow{\alpha_0, d_0} (v_1, \mathsf{s}^1) \ldots$. We denote by $\Omega^\tau(\mathcal{G}^t_D|\mathcal{A})$ all timed attacks of $\mathcal{G}^t_D|\mathcal{A}$. Let $\omega = v_0 d_0 \alpha_0, v_1 d_1 \alpha_1 \ldots$ be a timed attack, then we write $\omega \vDash^\tau t$ if there exists i, s.t. $[\![t]\!]v = \mathtt{tt}$ and $\sum_{i=0}^{i-1} d_i \leq \tau$.

Having introduced time, a defender may consider to not guarantee that an attack can never occur, but to make it very difficult time-wise i.e. that any succeeding attack will require more than τ time units - captured by question 2. Obviously, an attacker wishes to find an attack in response to $D \subseteq \mathtt{A}_d$ that succeeds before τ time units i.e. to answer question 3.

Question 2. For an attack-defense tree $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$, abstract timed attacker $\mathcal{A}$ and time limit τ, does there exist a $D \subseteq \mathtt{A}_d$, such that for all $\omega \in \Omega^\tau(\mathcal{G}^t_D|\mathcal{A})$, $\omega \nvDash^\tau t$?

Question 3. For an attack-defense tree $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$, abstract timed attacker $\mathcal{A}$, time limit τ and $D \subseteq \mathtt{A}_d$ does there exist $\omega \in \Omega^\tau(\mathcal{G}^t_D|\mathcal{A})$, such that $\omega \vDash^\tau t$?

2.2 Adding Stochasticity

A stochastic attacker is a tuple $\mathcal{A}^{\mathcal{S}} = (\mathcal{A}, \gamma, \{\delta_{p_a} | p_a \in \mathtt{A}_a^\lambda\})$, where $\mathcal{A}$ is an attacker defining allowed behaviour by the stochastic attacker, $\gamma : \mathsf{S} \to \mathtt{A}_a^\lambda \to \mathbb{R}_{\geq 0}$ assigns a probability mass to attacker's actions and for all $p_a \in \mathtt{A}_a^\lambda$,$\delta_{p_a} : \mathsf{S} \to \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ assigns a density to the execution time of p_a. A few requirements are in order here:

1. $\sum_{a \in \mathtt{A}_a^\lambda} \gamma(\mathsf{s})(a) = 1$,
2. $\int_{\mathbb{R}_{\geq 0}} \delta_a(\mathsf{s})(t)\,\mathrm{d}t = 1$ for all $a \in \mathtt{A}_a^\lambda$,
3. $\gamma(\mathsf{s})(a) \cdot \delta_a(\mathsf{s})(t) \neq 0$ implies $(a, t) \in Ac(\mathsf{s})$.

Requirement 1 states that $\gamma(\mathsf{s})$ must be a probability mass function, 2 requires that $\delta_a(\mathsf{s})$ is a probability density, and finally the most interesting rule 3 requires that whenever a probability density is assigned to a pair (a, t) then the attacker must in fact be able to do those according to the timed semantics. Finally, to make a complete stochastic semantics we need to resolve the non-determinism of selecting an outcome of performing an action p_a. We assume there is a static probability of an action succeeding, and thus we assume a probability mass function $\gamma_{Succ} : \mathtt{A}_a \to \{p_a, \neg p_a\} \to]0, 1[$ that assigns success and failure probabilities to actions with the requirement that any action must have a non-zero probability of succeeding.

Forming the core of a σ-algebra over timed attacks of $\mathcal{G}_D^t|\mathcal{A}^{\mathcal{S}}$, consider the finite sequence $\pi = v_0 I_0 \alpha_0 v_1 I_1 \alpha_1 \dots v_n$, where for all i; $\alpha_i \in \{p_a, \neg p_a \mid p_a \in \mathtt{A}_a\}$, I_i is an interval with rational end-points and $v_i \in \mathcal{V}$. The set of runs (cylinder) of this sequence is

$$\mathcal{C}_{\mathcal{G}_D^t|\mathcal{A}^{\mathcal{S}}}(\pi) = \{v_0 d_0 \alpha_0, v_1 d_1 \alpha_1 \dots v_n d_n \alpha_n \dots \in \Omega^\tau(\mathcal{G}_D^t|\mathcal{A}) \mid \forall i < d_i \in I_i\}.$$

The probability of these timed attacks runs from (v, s) are recursively defined by

$$F_{(v,\mathsf{s})}(\pi) = (v_0 = v) \cdot \gamma(\mathsf{s})(c(\alpha)) \cdot \int_{\mathbb{R}_{\geq 0}} \delta_{c(\alpha)}(\mathsf{s})(t)\,\mathrm{d}t \cdot \gamma_{Succ}(\alpha) F_{[(v,\mathsf{s})]^{\alpha,t}}(\pi^1),$$

where $\pi^1 = v_1 d_1 \alpha_1 \dots v_n d_n \alpha_n$, $c(p_a) = c(\neg p_a) = p_a$ and $(v, \mathsf{s}) \xrightarrow{\alpha,t} [(v, \mathsf{s})]^{\alpha,t}$ and base case $F_{(v,\mathsf{s})}(\epsilon) = 1$.

Remark 1. The stochastic semantics above is given for arbitrary time distributions. For the remainder we will however restrict our attention to stochastic attacker using only uniform distributions.

Let $\mathcal{G}_D^t = (\mathcal{V}, v^{\mathcal{A}}, \to_a, \to_\neg, F)$ be an AG and let $\mathcal{A}^{\mathcal{S}} = ((\mathsf{S}, M, Ac), \gamma, \{, \delta_{p_a} | p_a \in \mathtt{A}_a^\lambda\})$ then we let $F_{\mathcal{G}_D^t|\mathcal{A}^{\mathcal{S}}}(\pi) = F_{(v^{\mathcal{A}}, M(v^{\mathcal{A}}))}(\pi)$. With the above in place, the probability of a succesful attack within a time-bound τ is

$$\mathbb{P}_{\mathcal{G}_D^t|\mathcal{A}^{\mathcal{S}}}(\Diamond_{\leq \tau} t) = \int_{\omega \in \Omega^\tau(\mathcal{G}_D^t|\mathcal{A}^{\mathcal{S}})} \left(\begin{cases} 0 & \text{if } \omega \not\models^\tau t \\ 1 & \text{if } \omega \models^\tau t \end{cases} \right) \mathrm{d}F_{\mathcal{G}_D^t|\mathcal{A}^{\mathcal{S}}}.$$

Question 4. Given an attack-defense tree $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$, stochastic attacker $\mathcal{A}^{\mathcal{S}}$ and time limit τ; find $D^* = \arg\min_{D \in 2^{\mathtt{A}_d}} \left(\mathbb{P}_{\mathcal{G}^t_D | \mathcal{A}^{\mathcal{S}}}(\Diamond_\tau t) \right)$

Notice that question 4 has the time bound requirement for how quickly an attacker must succeed in an attack. If this time bound was not present and we thus gave an attacker unlimited time, then if a successful attack exists (no matter how unlikely) it would eventually succeed. This is evidenced by the plot in Fig. 3 with the time limit on the x-axis and the probabilities of an attack on the y-axis. The dashed line in the figure is the lower bound of the 99% confidence level and the solid line is the upper bound.

2.3 Adding Cost

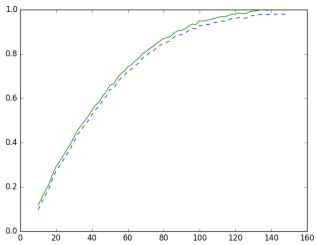


Fig. 3: Plot of probabilities of a successful attack for a uniform attacker.

Considering that an attacker is not only constrained by time, but also by his available resources e.g. money, we want to reflect the concept of a resource in our modelling. For this purpose we consider that an attacker only has one resource and that each action has an associated cost per attempted execution. We capture this cost by a function $\mathtt{C} : \mathtt{A}^{\lambda}_a \to \mathbb{R}_{\geq 0}$ that assigns the cost to actions with the requirement that $\mathtt{C}(\lambda) = 0$.

Let $\omega = v_0 d_0 \alpha_0 \ldots$ be a timed attack; then we define the cost of ω up till step j as $\mathtt{C}(\omega, j) = \sum_{i=0}^{j-1} \mathtt{C}(c(\alpha_i))$, where $c(\lambda) = \lambda$ and $c(p_a) = c(\neg p_a) = p_a$, i.e., we just sum up the individual costs along the attack before the j^{th} step. Now we can define the expected cost of a stochastic attacker, $\mathcal{A}^{\mathcal{S}}$, responding to a set of defense measures D with a time limit τ

$$\mathbb{E}_{\mathcal{G}^t_D | \mathcal{A}^{\mathcal{S}}}(C : \Diamond_{\leq \tau} t) = \int_{\omega \in \Omega^\tau(\mathcal{G}^t_D | \mathcal{A}^{\mathcal{S}})} \left(\begin{cases} \mathtt{C}(\pi, j) & \text{if } \omega \not\models^\tau t \wedge j = \max\{i \mid \sum_{k=0}^{i} d_k \leq \tau\} \\ \mathtt{C}(\pi, j) & \text{if } \omega \models^\tau t \wedge j = \min\{i \mid [\![t]\!] v_i = \mathtt{tt}\} \end{cases} \right) \mathrm{d}F_{\mathcal{G}^t_D | \mathcal{A}^{\mathcal{S}}}.$$

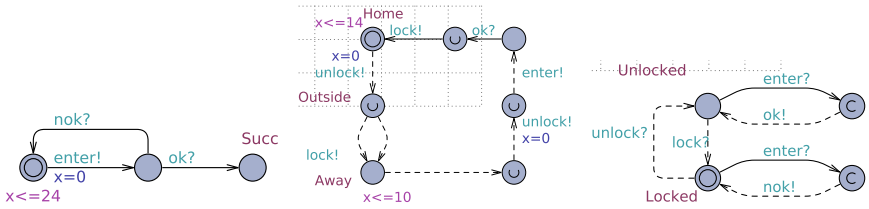
Question 5. Given an attack-defense tree $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$, stochastic attacker $\mathcal{A}^{\mathcal{S}}$, time limit τ and $D \subset \mathtt{A}_d$, find $\mathbb{E}_{\mathcal{G}^t_D | \mathcal{A}^{\mathcal{S}}}(C : \Diamond_{\leq \tau} t)$.

Consider that we fix the distribution over execution times and the success probabilities of execution attacks, but let γ range freely among all possible probability mass functions. Thus, we have a range of possible stochastic attackers, parameterised by γ, i.e. a range of attackers ${\mathcal{A}^{\mathcal{S}}}_1, {\mathcal{A}^{\mathcal{S}}}_2 \ldots$, where ${\mathcal{A}^{\mathcal{S}}}_i = (\mathcal{A}, \gamma_i, \{\delta_{p_a} | p_a \in \mathtt{A}^{\lambda}_a\})$. Then we are interested in finding the attacker that minimises the cost.

Question 6. Given an attack-defense tree $t \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$ time limit τ, $D \subset \mathtt{A}_d$ and a collection of attackers ${\mathcal{A}^{\mathcal{S}}}_1, {\mathcal{A}^{\mathcal{S}}}_2 \ldots$ parameterised by γ; find a stochastic attacker, $\mathcal{A}^{\mathcal{S}}$, minimising $\mathbb{E}_{\mathcal{G}^t_D | \mathcal{A}^{\mathcal{S}}}(C : \Diamond_{\leq \tau} t)$.

3 Timed Automata

In this paper we use the expressive *network of timed automata* (TA) formalism [1] extensively. An efficient model checking technique exists for this formalism, and the tool UPPAAL [5, 18] uses an extended version as its modelling language. As an example consider the three automata in Fig. 4, modelling two persons and a door.



(a) Thief (b) Resident (c) Door

Fig. 4: Model of a Thief, a Resident and a Door.

One of the persons is a Resident of a house and the other is a Thief who wants to enter the house while the Resident is not home. The Resident is initially at Home with the door locked for 14 hours - indicated by the expression $x <= 14$. The expression $x <= 14$ is an invariant expression and is something that should always be true whenever the automata is in the given location. From Home the resident may *unlock*! the door and go Outside, from where he can either *Lock*! the door or just leave the location to go Away. The "U" in Outside means this location is *urgent* and thus no time may pass while any automata is in such a location. The Door is initially Locked – from here someone may request to *enter*?, after which the Door responds with *ok*!: the "C" in the location means *committed* and is similar to urgent locations, but in addition to stopping time, it also ensures that only components in committed locations may move next. The door may be *lock*?ed - from which it responds to an *enter*? with a *nok*!. The Thief chooses some time, between 0 and 24 to attempt *enter*!ing – if he succeeds and gets an *ok*? from the Door he is happy and enters Succ. In case he is unlucky he receives an *nok*? and tries again later. Although simple, the above example contains the key elements of timed automata. To summarise, a timed automaton consists of locations and edges between locations. On locations one can write invariant expressions based on the values of clocks, like $x <= 14$. A clock is a real-valued counter that increases as time progresses. While moving along an edge, a TA may synchronise with another over a set of channels: in UPPAAL the convention is that *a*! means "send on a", and *a*? means "receive on a". Not shown in the example is that edges can be "guarded" by expressions over clocks.

Let c be a clock then we call an element $c \leq n$ ($c \geq n$) an upper (lower) bound and denote by $\mathcal{B}^{\leq}(\mathcal{C})$ ($\mathcal{B}^{\geq}(\mathcal{C})$) the set of all finite conjunctions of lower

(upper) bounds. For a finite set of channels Σ we denote by $\Sigma_o = \{a!|a \in \Sigma\}$ and $\Sigma_i = \{a?|a \in \Sigma\}$ the output and input actions over Σ respectively.

Definition 4 (Timed Automaton). *A timed automaton (TA) is a 6-tuple* $\mathcal{A} = (L, \mathcal{C}, \ell_0, \mathtt{A}_{,} \rightarrow, I)$*, where 1)* L *is a finite set of locations, 2)* $\ell_0 \in L$ *is the initial location, 3)* $\mathcal{C}$ *is a finite set of clocks, 4)* Σ *is a finite set of channels, 5)* $\rightarrow \subseteq L \times \mathcal{G}(\mathcal{C}) \times 2^{\mathcal{C}} \times L$ *is the (non-deterministic) transition relation. We write* $\ell \xrightarrow{g,a,R} \ell'$ *for a transition, where* ℓ *is the source and* ℓ' *the target location,* $g \in \mathcal{B}^{\leq}(\mathcal{C})$ *is a guard,* $a \in \Sigma_o \cup \Sigma_i$ *is a label, and* $R \subseteq \mathcal{C}$ *is the set of clocks to reset, and 6)* $I \colon L \rightarrow \mathcal{B}^{\geq}(\mathcal{C})$ *is an invariant function, mapping locations to a set of invariant constraints.*

A clock valuation is a function $v : \mathcal{C} \rightarrow \mathbb{R}_{\geq 0}$. We denote all clock valuations over $\mathcal{C}$ with $\mathcal{V}(\mathcal{C})$. We need two operations on clock valuations: $v' = v + d$ for a *delay* of $d \in \mathbb{R}_{\geq 0}$ time units, s.t. $\forall c \in \mathcal{C} \colon v'(c) = v(c) + d$, and *reset* $v' = v[R]$ of a set of clocks $R \subseteq \mathcal{C}$, s.t. $v'(c) = 0$ if $c \in R$, and $v'(c) = v(c)$ otherwise. We write $v \vDash g$ to mean that the clock valuation v satisfies the clock constraint g.

The semantics of a TA $(L, \mathcal{C}, \ell_0, \mathtt{A}_{,} \rightarrow, I)$ is a timed transition system with states $L \times \mathcal{V}(\mathcal{C})$ and initial state (ℓ_0, v_0), where v_0 assigns zero to all clocks. From a state (ℓ, v) the TA may transit via a discrete transition $(\ell, v) \xrightarrow{a} (\ell', v')$ if there exists an edge $\ell \xrightarrow{g,a,r} \ell'$, $v \vDash g$ and $v' = v[r]$. Time-wise the TA can perform a delay $d \in \mathbb{R}_{\geq 0}$ via a time transition $(\ell, v) \xrightarrow{d} (\ell, v + d)$ if $v + d \vDash I(\ell)$.

Several TAs $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$, $\mathcal{A}_i = (L_i, \mathcal{C}_i, \ell_0^i, \Sigma, \rightarrow_i, I_i)$ may be joined into a network of timed automata. The state space of such a composition is the product of the individual TAs state spaces. From a state $(s_1, s_2, \ldots, s_n)$ the network can do a

- discrete output transition $(s_1, s_2, \ldots, s_n) \xrightarrow{a!} (s'_1, s'_2, \ldots, s'_n)$, if there exists an i, such that $s_i \xrightarrow{a!} s'_i$ and for all $j \neq i$ $s_j \xrightarrow{a?} s'_j$
- or it can can delay d time units, $(s_1, s_2, \ldots, s_n) \xrightarrow{d} (s'_1, s'_2, \ldots, s'_n)$, if for all i $s_i \xrightarrow{d} s'_i$.

Notice we are using broadcast synchronisation for accommodating the use of UPPAAL SMC. Furthermore, we will assume that components are input-enabled and action-deterministic thus for any action there is at most one successor and for any input action there is at least one.

Stochastic Semantics The stochastic semantics of networks of timed automata was laid out by David et al. [8]. In a state, each timed automaton is given a delay density and a probability mass function for selecting output actions. The semantics is now race based: components select a delay, t, according to their delay distribution, and the one with the smallest delay is selected the winner. After the entire network performs the delay, the winner selects an output according to its probability mass function. The remaining network respond to this output by performing the corresponding input. Afterwards a new race commences. In UPPAAL SMC bounded delays (i.e. the current location has an invariant) are

selected from a uniform distribution ranging from the minimal delay before some guard is satisfied and the maximal delay, where the invariant is still satisfied. For unbounded delays the delay is selected from an exponential distribution.

In the preceding example, the probability that the Thief enters the house without the Resident being home within 12 time units is:

$$\int_0^{12} \frac{1}{14} \cdot \left(\int_t^{24} \frac{1}{24} \, \mathrm{d}t' \right) \cdot \frac{1}{2} \cdot \int_0^{12-t} \frac{1}{24-t} \, \mathrm{d}\tau \, \mathrm{d}t \approx 0.13$$

Game Semantics In recent works [9, 10] the simple stochastic timed automata model has been given a game semantics. In this semantics the edges of timed automaton $\mathcal{A} = (L, \mathcal{C}, \ell_0, \mathtt{A}, \rightarrow, I)$ are partitioned into a controllable set of edges, $\rightarrow_C$, and uncontrollable set of edges $\rightarrow_U$. The uncontrollable edges are controlled by stochastic environment behaving according to the stochastic semantics above, while the controllable set of edges is controlled by an actor that tries to "drive" the system into a given goal state. In Fig. 4 the dashed edges correspond to uncontrollable edges and the controllable edges are the solid edges.

A tool like Uppaal-Stratego can, by using reinforcement learning, find deterministic strategies for minimising the expected time (or cost) of reaching a goal - taking the stochastic environment into account.

4 Timed Automata Encoding

The timed automata encoding of the attack-defense tree semantics given in the previous sections consists of three automata; one encoding the attacker, one encoding the defender and one encoding the environment selecting an outcome for the execution of attacker actions (γ_{Succ}). Furthermore, the encoding has one boolean variable b_p_a per atomic proposition, p, in the attack-defense tree. The state of these boolean variables directly corresponds to the states of the ADG.

4.1 Environmental Modelling

Let $\mathtt{A}_a$ be the set of attacker actions in the attack-defense-tree, then for each $p_a \in \mathtt{A}_a$ we create a channel c_p_a that is used by the attacker to indicate that he wishes to execute p_a. The environment responds to this by deciding an outcome in accordance with γ_{Succ}. Fig. 5 depicts the modelling of the environment for an attack-defense tree, where there is only one attacker action (p_a); here $1 - p$ is the probability that p_a succeeds.

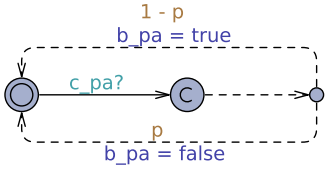


Fig. 5: Environmental modelling. In the figure p = 1-$\gamma_{Succ}(p_a)(p_a)$.

4.2 Defender Modelling

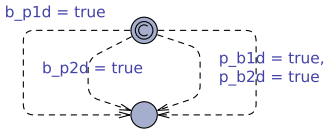


Fig. 6: Modelling the environment with two defender actions, p_d^1 and p_d^2.

Let $\mathtt{A}_d$ be the set of defender actions available to the defender. For each $D \in 2^{\mathtt{A}_d}$ the defender has an edge, where he sets all boolean variables, $p_d \in D$, to true. In Fig. 6 an example modelling of this is shown with two defender actions. As the edges of this defender are uncontrollable, the defender would select a set of defense measures by a uniform choice among all the edges. For analysing possible attack scenarios in response to a specific set of defense measures D we would delete edges of the defender until only the edge corresponding to D remains.

4.3 Attacker Modelling

In the formal development of an attacker we just defined general requirements that any attacker should respect. Firstly, we present a non-deterministic attacker that is as general as possible, which can be used for learning; afterwards we create one specific attacker profile, where the non-determinism is resolved by a probability mass function.

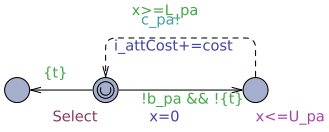


Fig. 7: Non-deterministic attacker modelling

Non-deterministic Attacker Assume we have $\mathtt{A}_a = \{p_a\}$ as our set of attacker actions and let each of the attacker propositions have a lower execution bound (L_pa) and an upper execution bound (U_pa) – an execution time that is not controllable by the attacker and thus will be selected according to a uniform distribution by the environment. Fig. 7 depicts an attacker with only one action: from the initial state, the attacker can decide to perform p_a, if it is not already true and the tree is not already true ($!\{t\}$); after which it enters a location, where the environment decides how long the execution takes according to the uniform distribution. After this waiting time the environment is informed of the attempt to execute p_a and decides on the outcome. Also, during this transition the cost of executing p_a is added to the variable $i\_attCost$. For the case with several propositions, the cycle in Fig. 7 is added for each proposition.

In case the tree is true, the attacker only has one option, namely, to enter the location, where he cannot do anything.

Uniform Attacker The uniform attacker is essentially the non-deterministic attacker, where the non-determinism of selecting an action is resolved by a uniform choice among all possible actions.

5 Tool Support

	L	U	C	F
is	0	20	80	0.80
bs	0	20	100	0.70
t	0	20	700	0.70
b	0	20	700	0.70
st	0	20	50	0.50
ba	0	20	85	0.60
im	0	20	70	0.50
ot	0	20	0	0.60

Fig. 8: Experimental setup.

The translation into timed automata described in the preceding section has been implemented as a python script. This script takes the attack-defense tree, the description of the cost of atomic attacker actions, the execution time and their probability of succeeding. Having translated into timed automata, we can now take advantage of the UPPAAL [5] model checking engine to answer some of the questions raised in the previous sections. For instance, question 3 is answerable by a simple timed reachability check by UPPAAL. In the following we focus on answering question 5 and question 6. We consider the attack-defense tree in Fig. 1. The success probability and cost of the various attacker actions are summarised in Fig. 8. The L column is the lower bound of the execution time, U is the upper bound, C is the cost of the actions and F is the success probability.

5.1 Expected Cost

We first show how to answer question 5 by finding expected cost of the uniform attacker within 300 time units. The cost of the attacker is estimated by the UPPAAL SMC with the query

$$\texttt{E}[<= 300; 1000](\texttt{max} : \texttt{i\_attCost}).$$

Defenses	$\{t1,tf\}$ $\{at,t2\}$	$\{tf,at\}$ $\{t2\}$	$\{t1,at\}$ $\{t2\}$	$\{at\}$ $\{t2\}$	$\{t1,tf\}$ $\{t2\}$	$\{tf\}$ $\{t2\}$	$\{t1\}$ $\{t2\}$	$\{t2\}$	$\{t1,tf\}$ $\{at\}$	$\{tf\}$ $\{at\}$	$\{t1\}$ $\{at\}$	$\{at\}$	$\{t1\}$ $\{tf\}$	$\{tf\}$	$\{t1\}$	$\emptyset$
Uniform	1030.31 ±33.20	873.16 ±28.68	880.81 ±29.0	891.17 ±29.7	1027.29 ±33.3	881.74 ±37.7	881.68 ±28.7	894.91 ±37.1	867.83 ±32.7	747.09 ±26.82	742.3 ±25.4	738.0 ±26.23	7730 ±31.35	659.19 ±27.9	659.4 ±27.1	675.6 ±27.64
UPPAAL-STRATEGO	855.86 ±31.6	292.76 ±13.2	256.11 ±6.4	256.25 ±6.6	876.12 ±31.4	242.62 ±5.8	246.02 ±5.9	267.14 ±7.5	498.58 ±32.3	240.22 ±8.62	183.32 ±7.9	197.310 ±8.9	304.14 ±31.1	219.13 ±17.43	173.95 ±8.74	115.03 ±7.10

Table 1: Expected cost for the uniform attacker and for synthesised strategies.

The estimates for various defense measures are given in Table 1 in the "Uniform" row. From the results we can see that the highest cost (unsurprisingly) is obtained when all possible defender's actions are selected, and the smallest when none of them are selected. Also the results indicate that by performing $t2, t1, tf$ the expected cost is equivalent to performing all of the defense measures. This is because this set jointly blocks large parts of the attack-defense tree, leaving only the expensive "threaten" and " blackmail" for an attacker to succeed.

5.2 Finding Good Attacker Profile

Next we answer question 6 i.e. we focus on a stochastic attacker, who minimises his costs in response to various defense measures. For doing this, we apply the non-deterministic attacker of our encoding and use the UPPAAL-STRATEGO to minimise the cost variable. The queries for the UPPAAL-STRATEGO are

$$\texttt{strategy s} = \texttt{minE}(\texttt{bi\_attCost})[<= 300] :<> \texttt{t}$$
$$\texttt{E}[<= 300; 1000](\texttt{max} : \texttt{bi\_attCost})\ \texttt{under s},$$

where t is the attack-defense tree translated into the UPPAAL syntax.

The result of executing these queries for different defenders are reported in Table 1 in the UPPAAL-STRATEGO row. As can be seen, the synthesized attacker generally obtains a reduced expected cost. The reason is that he can avoid attempting attacks he knows are blocked due to the defender's measures. Another reason is that this attacker actively attempts to minimise his costs; meaning he will not take the expensive "threaten" or "bribe" if it can be avoided.

6 Conclusion

In this paper we have shown how to separate the modelling of attacker's and defender's behaviours from the attack-defense tree. In this way we allow modelling complex temporal behaviours without compromising the intuitively simple description of various ways of achieving an attack expressed in the attack-defense tree. This stands in opposition with, for example, the work [12] that adds temporal behaviours by introducing sequential gates, trigger gates and reset gates, which may clutter the description of possible attacks. Experiments reported in the paper have shown the different analyses that can be performed on our encoding using the UPPAAL SMC and UPPAAL-STRATEGO: among these are finding an attacker who minimises his costs, and estimating the probability of an attack for a specific attacker.

In the future we wish to extend the current framework by describing the actual behaviour of the attacker in a more thorough way. This may include incorporating parts of the work by Hermanns et al. [12], but we will maintain them in a separate modelling language.

References

[1] R. Alur and D. L. Dill. Automata for modeling real-time systems. In M. Paterson, editor, *ICALP*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990. ISBN 3-540-52826-1.

[2] F. Arnold, D. Guck, R. Kumar, and M. Stoelinga. Sequential and parallel attack tree modelling. In *Computer Safety, Reliability, and Security*, pages 291–299. Springer, 2015.

[3] Z. Aslanyan and F. Nielson. Pareto Efficient Solutions of Attack-Defence Trees. In *Principles of Security and Trust*, volume 9036, page 95, 2015. doi:10.1007/978-3-662-46666-7_6.

[4] A. Bagnato, B. Kordy, P. H. Meland, and P. Schweitzer. Attribute decoration of attack–defense trees. *International Journal of Secure Software Engineering (IJSSE)*, 3(2), 2012.

[5] G. Behrmann, A. David, and K. G. Larsen. A tutorial on Uppaal. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004. doi:10.1007/978-3-540-30080-9_7.

[6] A. Buldas, P. Laud, J. Priisalu, M. Saarepera, and J. Willemson. Rational choice of security measures via multi-parameter attack trees. In *Critical Information Infrastructures Security*, pages 235–248. Springer, 2006.

[7] G. C. Dalton, R. F. Mills, J. M. Colombi, R. A. Raines, et al. Analyzing attack trees using generalized stochastic petri nets. In *Information Assurance Workshop, 2006 IEEE*, pages 116–123. IEEE, 2006.

[8] A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, J. van Vliet, and Z. Wang. Statistical model checking for networks of priced timed automata. In *FORMATS*, volume 6919 of *LNCS*, pages 80–96, 2011.

[9] A. David, P. G. Jensen, K. G. Larsen, A. Legay, D. Lime, M. G. Sørensen, and J. H. Taankvist. On time with minimal expected cost! In F. Cassez and J. Raskin, editors, *Automated Technology for Verification and Analysis*, volume 8837 of *Lecture Notes in Computer Science*, pages 129–145. Springer, 2014. ISBN 978-3-319-11935-9. doi:10.1007/978-3-319-11936-6_10.

[10] A. David, P. G. Jensen, K. G. Larsen, M. Mikucionis, and J. H. Taankvist. Uppaal stratego. In C. Baier and C. Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 9035 of *Lecture Notes in Computer Science*, pages 206–211. Springer, 2015. ISBN 978-3-662-46680-3. doi:10.1007/978-3-662-46681-0_16.

[11] O. Gadyatskaya. How to generate security cameras: Towards defence generation for socio-technical systems. In *Proc. of GraMSec*, volume 9390 of *LNCS*. Springer, 2015.

[12] H. Hermanns, J. Krämer, J. Krčál, and M. Stoelinga. The value of attack-defence diagrams. In *Principles of Security and Trust*, pages 163–185. Springer, 2016.

[13] M. G. Ivanova, C. W. Probst, R. R. Hansen, and F. Kammuller. Transforming graphical system models to graphical attack models. In *Proc. of GraMSec*, volume 9390 of *LNCS*. Springer, 2015.

[14] A. Jürgenson and J. Willemson. Computing exact outcomes of multi-parameter attack trees. In *On the Move to Meaningful Internet Systems: OTM 2008*, pages 1036–1051. Springer, 2008.

[15] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer. Attack–Defense Trees. *Journal of Logic and Computation*, 24(1):55–87, 2014.

[16] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer. DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. *Computer Science Review*, 13–14(0):1–38, 2014. doi:10.1016/j.cosrev.2014.07.001.

[17] R. Kumar, E. Ruijters, and M. Stoelinga. Quantitative Attack Tree Analysis via Priced Timed Automata. In *Formal Modeling and Analysis of Timed Systems*, pages 156–171. Springer, 2015.

[18] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *STTT*, 1 (1-2):134–152, 1997. doi:10.1007/s100090050010.

[19] S. Mauw and M. Oostdijk. Foundations of attack trees. In *Proceedings of the International Conference on Information Security and Cryptology (ICISC 2005)*, volume 3935 of *Lecture Notes in Computer Science*, pages 186–198. Springer, 2005.

[20] NATO Research and Technology Organisation (RTO). Improving Common Security Risk Analysis. Technical Report AC/323(ISP-049)TP/193, North Atlantic Treaty Organisation, University of California, Berkeley, 2008.

[21] F. Nielson, Z. Aslanyan, and D. Parker. Quantitative verification and synthesis of attack-defense scenarios. To appear in CSF'2016.

[22] OWASP. CISO AppSec Guide: Criteria for managing application security risks, 2013.

[23] C. Salter, O. S. Saydjari, B. Schneier, and J. Wallner. Toward a secure system engineering methodology. In *Proceedings of the 1998 New Security Paradigms Workshop (NSPW'98)*, pages 2–10, Charlottesville, Virginia, United States, September 1998.

[24] B. Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal*, December 1999.

[25] SITEC. Burglar resistance `https://www.sitec.de/en/information-and-advice/burglar-resistance/`.